Ratingiranje pri uglaševanju šahovskega programa z algoritmom diferencialne evolucije

Borko Bošković, Janez Brest, Aleš Zamuda, Viljem Žumer
Fakulteta za elektrotehniko, računalništvo in informatiko
Univerza v Mariboru
Smetanova ul. 17, 2000 Maribor, Slovenija
borko.boskovic@uni-mb.si

Rating System Inside the Tunig Process of a Chess Program with Differential Evolution Algorithm

This paper presents a rating system that was used for the tuning of a chess program inside differential evolution algorithm. The main challenge when tuning an already strong chess program is how to separate better and worse individuals. To solve this problem all played games during the evolutionary proces were stored and rating of individuals was calculated. Rating of individuals was used in the tuning process with injection mechanism and evaluation of individuls. This approach was tested with the tuning of a chess program BBChess. The tuned individual improved its playing strength for more than 150 ELO points according to the best initial individual.

1 Uvod

Računalniški šah ima že dolgo zgodovino raziskav na področju umetne inteligence. Cilj raziskav je izboljšati igralno moč šahovskih programov. Ta cilj lahko dosežemo na več načinov. Eden od načinov je izboljšava uporabljenih algoritmov v programih. Te izboljšave programom omogočajo večjo hitrost delovanja, programi so zmožni odkrivati več znanja, preiskujejo pomembnejša vozlišča drevesa igre itn. Pri teh izboljšavah igra ključno vlogo razvijalec programa. Program dopolnjuje in izboljšuje ter preizkuša, ali se je moč igranja povečala. To delo je zelo zahtevno in časovno potratno. Moč programa lahko izboljšamo tudi s pomočjo izdelave posebej namenske, ali z uporabo boljše strojne opreme, ki je lahko zelo draga.

Šahovski programi vsebujejo mnogo parametrov, ki vplivajo na njihovo delovanje. Pri konvencionalnem razvoju šahovskih programov, vrednosti parametrom nastavi razvijalec intuitivno. Nato preizkusi vpliv njihovih vrednosti na igralno moč. Ta postopek ponavlja določeno število iteracij. Ker je postopek preizkušanja dolgotrajen, je razvijalec omejen le na nekaj ponovitev. To je razlog, da vrednosti parametrov s pomočjo konvencionalnega razvoja ni možno nastaviti na optimalne vrednosti.

Za razliko od konvencionalnega razvoja lahko vrednosti parametrom nastavimo s pomočjo avtomatskega uglaševanja oz. razvijalca nadomestimo z računalnikom. V ta namen lahko uporabimo različne metode umetne inteligence. V tem članku se bomo osredotočili na algoritem diferencialne evolucije (DE), ki smo ga predstavili v [1].

Pri uglaševanju programa, kjer so parametri že relativno dobro nastavljeni, je težko določiti, kateri so boljši in kateri slabši. V ta namen smo v procesu uglaševanja odigrane igre shranjevali in na osnovi njih določali rating posameznikov. Dobljene ratinge smo uporabili v procesu ovrednotenja posameznikov in v mehanizmu vbrizgavanja. Ta način ocenjevanja posameznikov smo preizkusili pri uglaševanju našega šahovskega programa, katerega igralno moč smo izboljšali za več kot 150 ELO točk.

V drugem poglavju predstavimo metodo uglaševanja. Nato v tretjem in četrtem poglavju opišemo eksperiment in dobljene rezultate. V petem poglavju podamo kratek zaključek.

2 Algoritem uglaševanja

Algoritem uglaševanja temelji na algoritmu diferencialne evolucije, ki smo ga predstavili v [1]. Algoritem smo spremenili in dopolnili tako, da bolj merodajno ocenjuje boljše posameznike. Za razliko od že predstavljenega algoritma, tokrat nismo uporabili mehanizma adaptacije. Kontrolni parameter `F` se uporablja enako kot v klasičnem algoritmu DE [14, 13, 10, 5]. Ostale spremembe in dopolnila bodo opisana v nadaljevanju tega poglavja.

2.1 Ratingiranje

Ratingiranje omogoča, da glede na odigrane igre določimo moč igralcev. V že predstavljenih algorit-

mih za uglaševanje s pomočjo evolucijskih algoritmov [8, 6, 3, 9, 2, 1, 4] so bile uporabljene preproste metode za ocenjevanje posameznikov kot je npr. razmerje med številom zbranih točk in številom odigranih iger. Pri tem načinu ocenjevanja slabši posameznik lahko doseže boljšo oceno v primerjavi z boljšim posameznikom. Razlog temu je v izbiri nasprotnikov in številu odigranih iger. Če je posameznik odigral malo število iger, je nemogoče oceniti njegovo dejansko moč. Do napačne ocenitve lahko pride v primeru, kadar slabši posameznik igra igre nasproti v povprečju slabšim igralcem kot boljši posameznik. Tem anomalijam pri ocenjevanju posameznikov s podobno močjo se praktično ni mogoče izogniti. S pomočjo boljših načinov ocenjevanja posameznikov lahko metodo uglaševanja izboljšamo oz. zmanjšamo šum.

Algoritem 1 Način igranja iger

```
for i = 0 to NP do
  genRez_i = 0;
  {Igranje trenutne populacije}
  for n = 0 to igre do
    m = i;
    while m == i do
      m = rand(0,NP);
    end while
    if n%2 == 0 then
      rez = odigrajIgro(x^c_{g,i}, x^b_{g,m}, depth);
    else
      rez = 1 - odigrajIgro(x^b_{g,m}, x^c_{g,i}, depth);
    end if
    genRez_i = genRez_i + rez;
  end for
  {Igranje poskusne populacije}
  for n = 0 to igre do
    m = i;
    while m == i do
      m = rand(0,NP);
    end while
    if n%2 == 0 then
      rez = odigrajIgro(x^t_{g,i}, x^b_{g,m}, depth);
    else
      rez = 1 - odigrajIgro(x^b_{g,m}, x^t_{g,i}, depth);
    end if
    genRez_i = genRez_i - rez;
  end for
end for
```

Da bi zmanjšali šum pri ocenjevanju posameznikov, smo v proces uglaševanja vpeljali ratingiranje, ki se uporablja tudi za ratingiranje šahovskih igralcev na različnih turnirjih in lestvicah. Pri našem uglaševanju smo uporabili ratingiranje, ki temelji na "Minorization-Maximization" algoritmu [7] in je implementirano v orodju Bayeselo[1].

Za ratingiranje posameznikov smo v proces uglaševanja dodali podatkovno bazo vseh odigranih iger. Da bi dosegli veliko število odigranih iger med najbolje ocenjenimi posamezniki, smo spremenili tudi način igranja iger in vpeljali mehanizem vbrizgavanja. Vsak posameznik iz trenutne in poskusne populacije igra določeno število iger, nasproti naključno izbranim posameznikom med trenutno NP najbolje ratingiranimi posamezniki, kot prikazuje algoritem 1.

V algoritmu $\mathbf{x}^c_{g,i}$ predstavlja posameznika iz trenutne populacije, $\mathbf{x}^t_{g,i}$ posameznika iz poskusne populacije in $\mathbf{x}^b_{g,m}$ posameznika, ki pripada NP trenutno najbolje ratingiranim posameznikom. Kontrolni parameter igre določa število iger v eni generaciji. Vrednost tega parametra določa uporabnik. NP je kontrolni parameter oz. velikost populacije kot pri klasičnem algoritmu DE.

Ko se igranje iger v določeni populaciji konča, vse igre dodamo v skupno bazo iger, na osnovi katere nato izračunamo ratinge vseh posameznikov.

2.2 Selekcija

V algoritmu 1 je uporabljena spremenljivka $\mathtt{genRrz}_i$, na osnovi katere operator selekcije določa, kateri posamezniki bodo preživel v naslednjo populacijo:

$$\mathbf{x}^c_{g+1,i} = \begin{cases} \mathbf{x}^c_{g,i} & \mathtt{genRez}_i \geq 0 \\ \mathbf{x}^t_{g,i} & \text{drugače} \end{cases}$$

Prikazana selekcija ne temelji na ratingiranju. Igralci igrajo določeno število iger proti trenutno NP najbolje ratingiranim posameznikom. V naslednjo generacijo preživijo posamezniki, ki zberejo več točk kot njihovi pripadajoči nasprotniki. Za zmago igralec dobi 1 točko, za remi 0,5 točk in 0 za poraz. Selekcija je tako odvisna le od odigranih iger v trenutni generaciji in pripadajoča posameznika imata enako možnost za preživetje, ne glede na to kaj se je dogajalo v prejšnjih generacijah.

2.3 Vbrizgavanje

Z namenom, da bi dobri posamezniki še dodatno vplivali na izgradnjo novih posameznikov, smo uporabili še mehanizem vbrizgavanja. S pomočjo proporcionalne selekcije glede na doseženi rating, vbrizgamo določeno število posameznikov v poskusno populacijo kot prikazuje algoritem 2. Ta mehanizem uporabimo po operaciji križanja in mutacije.

V algoritmu 2 $\mathtt{B}_g$ predstavlja urejeno (padajoče) bazo vseh ratingiranih posameznikov. $\mathtt{C}_g$ je trenutna populacija in $\mathtt{T}_g$ je poskusna populacija. HI je kontrolni parameter, ki definira koliko posameznikov se bo poskusilo vbrizgati in določa ga uporabnik. Posameznike vbrizgavamo le v

[1] Bayeselo je prosto dostopno orodje za rangiranje igralcev glede na izračunane ratinge.

Algoritem 2 Vbrizgavanje

```
size = HS;
if HS > sizeOf(B_g) then
    size=sizeOf(B_g);
end if
for k = 0 to HI do
    x = propSelekcija(x^b_{g,0}, x^b_{g,1}, ..., x^b_{g,size-1});
    if not vsebuje(C_g, T_g, x) then
        m = rand(0, NP);
        x^t_{g,m} = x;
    end if
end for
```

primeru, kadar se ne nahajajo v trenutni ali poskusni populaciji.

3 Eksperiment

Opisan način uglaševanja smo preizkusili s pomočjo uglaševanja šahovskega programa BBChess 1.2. Kontrolne parametre algoritma smo nastavili, kot prikazuje tabela 1. Oznaka `D` prikazuje dimenzijo problema, v našem primeru število parametrov ocenitvene funkcije, ki jih uglašujemo. Oznaki `F` in `CR` označujeta kontrolna parametra mutacije in križanja. Oznaka `G` predstavlja število generacij za zagon uglaševanja. Izbrana je bila strategija DE/rand/1/bin, ki je ena od najbolj uporabljenih strategij algoritma DE. Oznaka `JR` je kontrolni parameter uporabljenega mehanizma nasprotij [12, 11, 15]. Oznaka `depth` pa določa globino iskanja šahovskega programa in `niti` število iger, ki se igra istočasno.

Tabela 1: Kontrolni parametri

NP	20	JR	0,1
D	190	HI	4
F	0,5	depth	6
CR	0,8	niti	2
G	100	igre	10
strategija	DE/rand/1/bin		

Parametre ocenitvene funkcije, ki smo jih uglaševali, smo nastavili intuitivno. V procesu uglaševanja je šahovski program uporabljal 16 MB pomnilnika za transpozicijsko tabelo in 16 MB pomnilnika za bazo končnic. V otvoritvenih fazah igre je program uporabljal otvoritveno knjižnico `performace.bin` avtorja Marca Lacrossea in v končnici baze končnic avtorja Daniela Shawtula. V začetni populaciji so parametri bili nastavljeni uniformno naključno znotraj določenih intervalov. Eksperiment smo zagnali na računalniku z Intel Core 2 CPE 2,4 GHz in 1 GB RAM. Operacijski sistem je bil Linux Mandriva 2008 x86_64. Program je bil preveden z O3 optimizacijami za x86_64 arhitekturo s pomočjo prevajalnika GCC 4.2.2. Eksperiment je trajal 50 ur.

4 Rezultati

Da bi ugotovili, kakšen je rezultat uglaševanja oz. izboljšava ocenitvene funkcije, smo izbrali najboljšega ratingiranega posameznika (Neuglašen) iz inicializacijske populacije in 20 najbolje ratingiranih posameznikov (Uglašen) na koncu procesa uglaševanja. Za 20 najbolje ratingiranih posameznikov smo nato izračunali rating, tako da so igrali igre nasproti šahovskemu programu Rybka z omejeno močjo 1700 ELO, 2100 ELO in 2300 ELO. Dobljeni rezultati so prikazani v tabeli 2. Prvi stolpec v tabeli predstavlja rangiranje igralcev po doseženem ratingu. Drugi stolpec je ime posameznika in tretji stolpec doseženi rating. Četrti in peti stolpec določata interval, na katerem se nahaja dejanski rating igralca s 95% verjetnostjo. Npr. najbolje rangirani med uglaševanimi posamezniki je dosegel rating s 95% verjetnostjo na intervalu $[2068-48, 2068+48]$. Šesti stolpec prikazuje število odigranih iger, sedmi stolpec povprečni rating nasprotnika in zadnji stolpec delež remijev.

Tabela 2: Ratingiranje posameznikov glede na Rybko

Št.	Ime	ELO	+	-	Igre	Rez.	Nas.	Remi
1	Rybka 2300	2277	23	23	1000	83%	1996	13%
2	Rybka 2100	2118	20	20	1000	66%	1996	13%
3	Uglašen 1	2068	48	48	200	55%	2020	9%
4	Uglašen 2	2067	47	47	200	55%	2020	12%
5	Uglašen 3	2061	47	47	200	54%	2020	13%
6	Uglašen 4	2042	47	47	200	53%	2020	17%
7	Uglašen 5	2038	47	47	200	52%	2020	12%
8	Uglašen 6	2034	47	47	200	52%	2020	13%
9	Uglašen 7	2029	46	46	200	50%	2020	16%
10	Uglašen 8	2023	46	46	200	51%	2020	21%
11	Uglašen 9	2016	48	48	200	50%	2020	10%
12	Uglašen 10	2007	48	48	200	48%	2020	6%
13	Uglašen 11	2001	46	46	200	48%	2020	13%
14	Uglašen 11	1983	48	48	200	46%	2021	11%
15	Uglašen 12	1975	47	47	200	45%	2020	12%
16	Uglašen 13	1974	47	47	200	44%	2020	13%
17	Uglašen 14	1974	48	48	200	44%	2020	14%
18	Uglašen 15	1960	48	48	200	42%	2020	16%
19	Uglašen 16	1951	48	48	200	42%	2020	11%
20	Uglašen 17	1940	48	48	200	41%	2020	13%
21	Rybka 1900	1935	19	19	1000	42%	1996	15%
22	Uglašen 18	1897	47	47	200	36%	2020	17%
23	Uglašen 19	1882	50	50	200	35%	2020	11%
24	Rybka 1700	1749	23	23	1000	21%	1996	10%

Najbolje rangiran uglašen posameznik, neuglašen posameznik in dva zelo znana odprto-kodna programa Crafty-21.6 in GNU Chess 5.07, so odigrali turnir. Na osnovi rezultatov tega turnirja (tabela 3) lahko vidimo, da se je uglašeni program oz. ocenitvena funkcija izboljšala za **164 ELO** točk. Iz rezultatov je razvidno tudi, da so intuitivno nastavljeni intervali parametrov bili dobri. Ne-

uglašen posameznik, ki je imel naključno izbrane parametre se je izkazal kot boljši igralec od programa GNU Chess. Ratingi, ki so jih dosegli igralci, so bili izračunani na osnovi ratinga programa Crafty, ki ima na CCRL[2] lestvici rating 2647 ELO.

Tabela 3: Turnir - 5 minut za 40 potez

št	Ime	ELO	+	-	Igre	Rez.	Nas.	Remi
1	Crafty-21.6	2647	52	52	300	90%	2240	10%
2	Uglašen 1	2406	41	41	300	59%	2320	11%
3	Neuglašen	2242	43	43	300	36%	2375	10%
4	GNU Chess 5.07	2071	50	50	300	16%	2432	4%

5 Zaključek

V članku smo predstavili algoritem uglaševanja šahovskega programa, ki uporablja sistem ratingiranja za ocenjevanje posameznikov in prilagojen način igranja iger. Tako dobri posamezniki odigrajo veliko števil iger, na osnovi katerih v procesu uglaševanja bolje rangiramo posameznike. V algoritem smo vključili še mehanizem vbrizgavanja posameznikov, ki dobre posameznike, čeprav izumrejo, vrača v proces uglaševanja.

Predstavljen algoritem smo preizkusili s pomočjo uglaševanja šahovskega programa BBChess 1.2. Program smo uglaševali skozi 100 generacij. Na koncu preizkusa smo dobili uglašen program, ki smo ga primerjali z najboljšim posameznikom iz inicializacijske populacije. Programa smo primerjali tako, da sta igrala turnir skupaj z odpro-kodnima programoma GNU Chess in Crafty.

Iz rezultatov je razvidno, da se je uglašeni program izboljšal za **164 ELO** točk in dosegal rating **2406 ELO**. Neuglašeni posameznik je dosegel večji rating od programa GNU Chess, kar pomeni, da smo uspešno uglasili že dokaj zmogljiv šahovski program. Tako si pri razvoju šahovskih programov lahko s pomočjo predstavljenega algoritma razvijalci uglasijo parametre in povečajo igralno moč programov.

Literatura

[1] B. Bošković, J. Brest, A. Zamuda in V. Žumer. Uglaševanje šahovskega programa BBChess z uporabo algoritma diferencialne evolucije. *Zbornik šestnajste mednarodne Elektrotehniške in računalniške konference ERK 2007, zv. B*, str. 73–76, 2007.

[2] B. Bošković, S. Greiner, J. Brest in V. Žumer. A Differential Evolution for the Tuning of a Chess Evaluation Function. *The 2006 IEEE Congress on Evolutionary Computation CEC 2006*, str. 6742–6747. IEEE Press, 2006.

[3] B. Bošković, S. Greiner, J. Brest in V. Žumer. Adaptivni algoritem diferencialne evolucije za uglaševanje parametrov ocenitve funkcije računalniškega šaha. *Zbornik petnajste mednarodne Elektrotehniške in računalniške konference ERK 2006*, str. 83–86, 2006.

[4] B. Bošković, S. Greiner, J. Brest, A. Zamuda in V. Žumer. An Adaptive Differential Evolution Algorithm with Opposition-Based Mechanisms, Applied to the Tuning of a Chess Program. Uday K Chakraborty, urednik, *Advances in Differential Evolution, Studies in Computational Intelligence*, št. 143. Springer, 2008.

[5] Vitaliy Feoktistov. *Differential Evolution: Search of Solutions (Springer Optimization and Its Applications)*. Springer-Verlag New York, Inc, Secaucus, NJ, ZDA, 2006.

[6] David B. Fogel, Timothy J. Hays, Sarah L. Hahn in James Quon. A Self-Learning Evolutionary Chess Program. *Proceedings of the IEEE*, 92(12):1947–1954, 2004.

[7] David R. Hunter. MM algorithms for generalized Bradley-Terry models. *Annals of Statistics*, 32(1):384–406, 2004.

[8] G. Kendall in G. Whitwell. An Evolutionary Approach for the Tuning of a Chess Evaluation Function Using Population Dynamics. *Proceedings of the 2001 Congress on Evolutionary Computation CEC 2001*, str. 995–1002, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seul, Koreja, 2001. IEEE Press.

[9] H. Nasreddine, H.S. Poh in G. Kendall. Using an Evolutionary Algorithm for the Tuning of a Chess Evaluation Function Based on a Dynamic Boundary Strategy. *Proceedings of 2006 IEEE International Conference on Cybernetics and Intelligent Systems (CIS 2006)*, str. 1–6, 2006.

[10] K. V. Price, R. M. Storn in J. A. Lampinen. *Differential Evolution, A Practical Approach to Global Optimization*. Springer, 2005.

[11] S. Rahnamayan, H. R. Tizhoosh in M. M. A. Salama. Opposition-Based Differential Evolution Algorithms. *The 2006 IEEE Congress on Evolutionary Computation CEC 2006*, str. 7363–7370, 2006.

[12] S. Rahnamayan, H. R. Tizhoosh in M. M. A. Salama. Opposition-Based Differential Evolution. *IEEE Transactions on Evolutionary Computation*, 12(1):64–79, 2008.

[13] R. Storn in K. Price. Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, Berkeley, CA, 1995.

[14] R. Storn in K. Price. Differential Evolution - A Simple and Efficient Heuristic for Global Optimisation Over Continuous Spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[15] H. R. Tizhoosh. Opposition-Based Learning: A New Scheme for Machine Intelligence. *Proceedings of International Conference on Computational Intelligence for Modelling Control and Automation - CIMCA 2005*, str. 695–701, Dunaj, Avstrija, 2005.

[2] CCRL (Computer Chess Rating Lists) je šahovska lestvica šahovskih programov oz. klub ljudi, ki preizkušajo šahovske programe in jih primerjajo med seboj. Dostopno na: http://computerchess.org.uk/.